Technical Note: The Modular Earth Submodel System (MESSy) - a new approach towards Earth System Modeling

P. Jöckel, R. Sander, A. Kerkweg, H. Tost, J. Lelieveld

Atmos. Chem. Phys., 5, 433–444, 2005
www.atmos-chem-phys.org/acp/5/433/
SRef-ID: 1680-7324/acp/2005-5-433
European Geosciences Union

Atmospheric Chemistry and Physics

Technical Note: The Modular Earth Submodel System (MESSy) – a new approach towards Earth System Modeling

P. Jöckel, R. Sander, A. Kerkweg, H. Tost, and J. Lelieveld

Max-Planck-Institute for Chemistry, P.O. Box 3060, 55020 Mainz, Germany

Received: 17 September 2004 – Published in Atmos. Chem. Phys. Discuss.: 4 November 2004
Revised: 2 February 2005 – Accepted: 3 February 2005 – Published: 10 February 2005

Abstract. The development of a comprehensive Earth System Model (ESM) to study the interactions between chemical, physical, and biological processes, requires coupling of the different domains (land, ocean, atmosphere, ...). One strategy is to link existing domain-specific models with a universal coupler, i.e. an independent standalone program organizing the communication between other programs. In many cases, however, a much simpler approach is more feasible. We have developed the Modular Earth Submodel System (MESSy). It comprises (1) a modular interface structure to connect submodels to a base model, (2) an extendable set of such submodels for miscellaneous processes, and (3) a coding standard. MESSy is therefore not a coupler in the classical sense, but exchanges data between a base model and several submodels within one comprehensive executable. The internal complexity of the submodels is controllable in a transparent and user friendly way. This provides remarkable new possibilities to study feedback mechanisms (by two-way coupling). Note that the MESSy and the coupler approach can be combined. For instance, an atmospheric model implemented according to the MESSy standard could easily be coupled to an ocean model by means of an external coupler. The vision is to ultimately form a comprehensive ESM which includes a large set of submodels, and a base model which contains only a central clock and runtime control. This can be reached stepwise, since each process can be included independently. Starting from an existing model, process submodels can be reimplemented according to the MESSy standard. This procedure guarantees the availability of a state-of-the-art model for scientific applications at any time of the development. In principle, MESSy can be implemented into any kind of model, either global or regional. So far, the MESSy concept has been applied to the general circulation model ECHAM5 and a number of process box-models.

Correspondence to: P. Jöckel
(joeckel@mpch-mainz.mpg.de)

1 Introduction

A new approach in global environmental computer modeling is to pursue Earth System Models. The aim is to capture feedback mechanisms between the traditional components as defined within the geosciences, i.e. the atmosphere, hydrosphere, lithosphere, pedosphere, biosphere, and ultimately also the anthroposphere. In the past, many component models used pre-calculated data sets (offline models) to circumvent computational constraints. Since the large data sets involved call for large storage capacity, especially at high time resolution, a number of processes could only be updated occasionally. Data for intermediate periods were interpolated in time, and small-scale or rapidly proceeding processes were parameterized, leading to a loss of accuracy and flexibility. The coupling of such traditional system component models causes data storage capacity to become a limiting factor. Furthermore, the time resolution required for one component may not match that of the other. The obvious solution is to interactively compute all processes at relatively high and flexible time resolution, which reduces the need to store data, and which allows capturing interactions and feedbacks previously suppressed by interpolation or parameterization. On the other hand, more comprehensive and complex models are more difficult to handle. They require increasingly powerful computers and the application of software engineering techniques (Post and Votta, 2005). Our philosophy to pursue an interactively coupled Earth System model approach is partly based on the expectation that computational power will increase more rapidly than data storage and handling capacity. In this technical note we present the requirements, the outline and the implementation of our model structure.

Whereas in the early phase of the development most of the "historically grown" models have been designed to address a few very specific scientific questions in a specific geophysical domain, the codes have been continuously further developed over decades, with steadily increasing complexity. An increasing number of processes has been taken into

consideration, so that the computability was usually close to the limits of the available resources. These historically grown model codes are now in a state associated with several problems (see also Post and Votta, 2005):

- The code has mostly been developed by scientists, who are not necessarily well-trained programmers. In principle every contribution follows its developer's unique style of programming. Coding conventions (e.g. http://www.meto.gov.uk/research/nwp/numerical/fortran90/f90_standards.html) only help when strictly adhered and when code reviews are performed on a regular basis.
- The code has not been written to be easily extendable and adaptable to new scientific questions.
- There has been little motivation for writing quality (i.e. readable and robust) code, since the scientific aim (i.e. only the model output) had to be reached rapidly and uniquely. Well structured, readable code has usually received little priority.
- Documentation lines within the code are often rare or absent.
- The code has been developed to run in a few specific configurations only, e.g. in a particular vertical and horizontal resolution, and parameterizations are resolution dependent.
- The code contains "hard-coded" statements (e.g. parameters implemented explicitely as numerical values), which require recompilation after changes, e.g. for sensitivity studies.
- In many cases code developers (e.g. PhD students) are no longer available for support and advice. If insurmountable obstacles occur, the code has to be rewritten completely.
- Outdated computer languages (mostly Fortran77 or older) limit the full exploitation of available hardware capacities. Therefore, the codes have been "optimized" for specific hardware architectures, using non-standard, vendor-specific language extensions. As a consequence, these codes are not portable to other platforms.
- Compilers have been highly specific and error tolerant, e.g. some even allowed divisions by zero. Although this may seem an advantage, it must be stressed that potentially serious code flaws are masked, which makes error tracing difficult.

The result is often a highly importable, unreadable, undocumented "spaghetti-code", which inhibits an efficient further development. The same problems have to be solved time and again. The use of submodels/routines in a different environment requires in many cases incommensurate efforts. Even worse than this development aspect is the fact that those complex, non-transparent computer programs elude more and more understanding, apart from a small, indispensable group involved from the start.

These problems might call into question the feasibility of the next step, the transition from domain specific models of the Earth's Environment towards comprehensive Earth System Models (ESMs). One popular approach is to couple the existing domain specific models as they are via an external "coupler", which handles the communication (data-exchange) between the domain models (Fig. 1).

This approach is followed for instance by the PRogram for Integrated earth System Modeling (PRISM, http://prism.enes.org/) and the Earth System Modeling Framework (ESMF, http://www.esmf.ucar.edu/).

However, this concept might not be sufficient for the development of an understandable, traceable, complex ESM. Increasing complexity requires an equally increasing degree of transparency in the models. One single user, focusing on a specific scientific question, is unable to grasp the whole model setup. Still, she/he must be able to control it. For this also the domain specific models need some re-configuration ("cleanup"), as outlined in Fig. 2.

The base idea is to modularize the different specific processes, i.e. to implement them as submodels and separate them from the remaining base model. This is the consistent application of the operator splitting (Fig. 3), which is implemented in such models anyway.

In offline models operator splitting is a problem because the time resolution of some process calculations is low. In relatively high resolution online models this problem systematically decreases.

It must be stressed that both the domain-oriented approach, and the process-oriented approach are not exclusive. Both of those can be combined. For example, the domain specific models may be implemented following the process-oriented approach and then coupled with the domain-oriented approach using an external coupler. This combination allows a flexible, efficient, problem-oriented development of ESMs based on existing codes.

In Sect. 2 the needs for a successful Earth System Model are set out in more detail, and suggestions on how to meet these requirements are elaborated. The implementation of these ideas as the "Modular Earth Submodel System" (MESSy) is presented in Sect. 3. Finally, Sect. 4 suggests possible applications.

2 Objectives

Any Earth System Model in general must fulfill at least the following conditions to be consistent, physically correct, flexible, sustainable, and extendable:

- Flexibility: Several alternative implementations for the same process can coexist in the same model system, e.g.

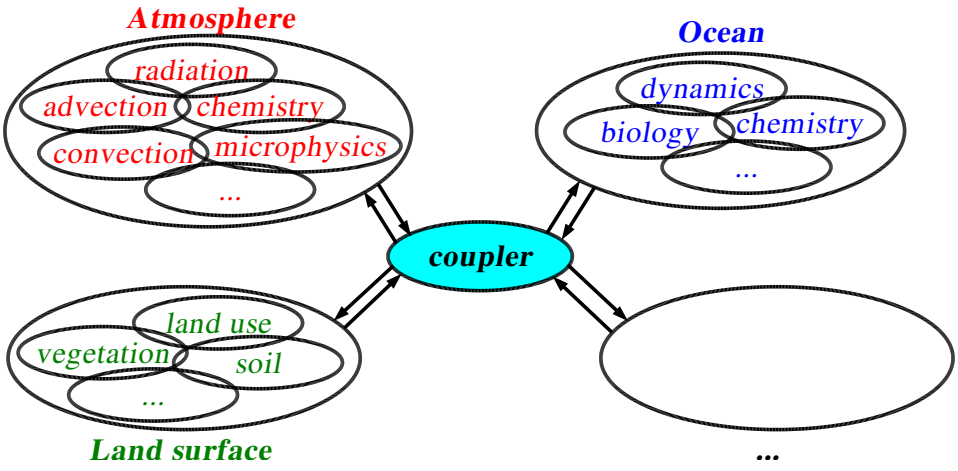

Fig. 1. Domain-oriented approach for building an ESM from existing domain specific models (atmosphere, ocean, land surface, ...). Data exchange can be controlled from each domain model and organized via a universal coupler. The domain models and the coupler are self-contained executables running simultaneously; communication is performed via the coupler.

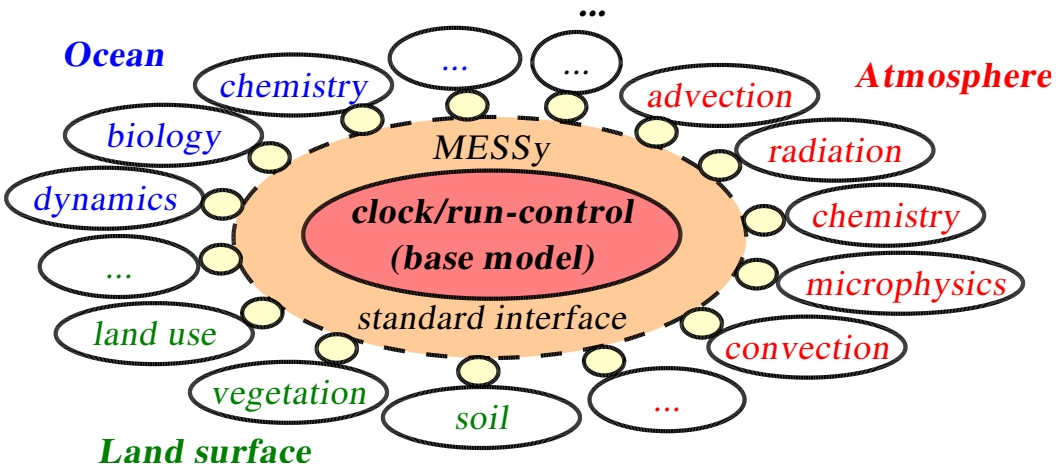

Fig. 2. Process-oriented approach to establish an ESM. Each physical process is coded as a modular entity connected via a standard interface to a common base model. The base model can be for instance an atmosphere or ocean GCM, etc. At the final development state, the base model contains hardly more than a central clock and a run control for all modularized processes. All processes and the base model together form one comprehensive executable. Via the standard interface, data exchange between all processes is possible.

different parameterizations for sensitivity studies, different approaches for process studies, "frozen" and developer versions can be directly compared.

– Plug & play: The implemented processes can be easily exchanged between different model systems.

– Test facility: The implemented processes can be tested without running the entire model system, i.e. for instance coupled to a simple box model (see Sect. 4.1).

– Security: Parameterizations can introduce large errors if they are used outside their valid range. Such unwanted or uncontrolled extrapolations should be avoided by terminating the model system if a parameterization gets out of range.

– Coupled system: Feedback mechanisms can be easily implemented and controlled. In addition, the transparent data flow offers new possibilities towards a systematic study and quantification of feedback mechanisms.

– Multi-purpose: The model system can be applied to a wide range of scientific questions, especially with respect to spatial and temporal scales, the processes involved, and the domains covered.

– Portability: The model system is portable and runs on various different computer architectures.

– Expandability: The model system structure allows the straightforward adaption of additional processes and is prepared for future contributions.

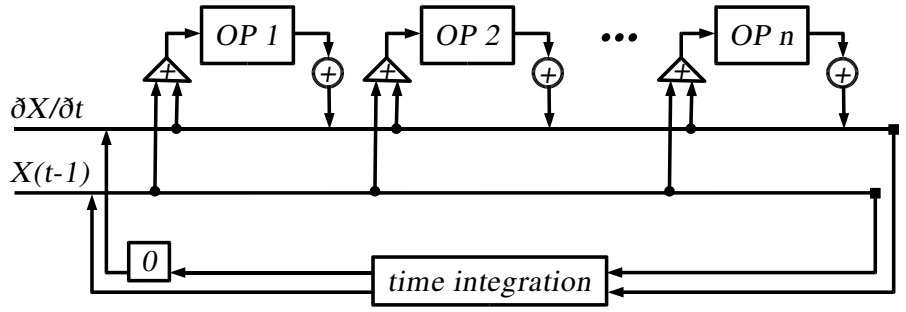

Fig. 3. Operator splitting time integration scheme (in this case a second order scheme used in many GCMs). Each process is represented by an operator (OP 1, OP 2, ..., OP n) which calculates a tendency ($\partial X/\partial t$) for the quantity X based on the quantity X at the time step before ($t-1$) and the sum of all tendencies calculated by the operators in the sequence before.

– Multi-developer: The model system can be further developed by more than one person at the same time without interference.

– Consistency: The model system is consistent, all implemented processes share the same fundamental data sources.

– Efficiency: The model system code is efficient regarding usage of computer (processor) time.

– Reproducibility: Re-compilation of the code is avoided whenever possible, at least within one model simulation including sensitivity studies. Especially the choice of process specific parameters, the coupling of different processes, and the choice of available alternatives should not require a code recompilation. Note: In coupled complex (non-linear) systems, re-compilation bears the risk of loosing reproducibility due to uncontrollable compiler issues.

– Variable complexity: The degree of complexity of each process can be changed according to its relevance in different applications.

– Synergy: Implementations relevant for different processes are shared.

– User friendly: The model system comprises a unified, transparent user interface for the control of the model system.

To reach these aims simultaneously, we define the following framework for the specific implementation of MESSy:

– Modularity: Each specific process is coded as a separate, independent entity, i.e. as a submodel, which can be switched on/off individually.

– Standard interface: A so-called base model provides the framework to which all submodels are connected. At the final state of development the base model should not contain more than a central clock for the time control (time integration loop) and a flow control of the involved processes (=submodels). This ultimate aim can be reached stepwise. For instance one could start from an existing GCM (as in our example) and connect new processes via the standard interface. At the same time, it is possible to modularize processes which are already part of the GCM, and reconnect them via the standard interface. In many cases this requires only a slightly modified reimplementation based on the existing code.

– Self-consistency: Each submodel is self-consistent, the submodel output is uniquely defined by its numerical input.

– Resolution independency: The submodel code is independent of the spatial (grid) and temporal resolution (time step) of the base model. If applicable and possible, the submodels are also independent of the dimensionality (0-D (box), 1-D (column), 2-D, 3-D) and the horizontal (regional, global) and vertical domain of the base model. Each process is coded for the smallest applicable entity (box, column, column-vector, ...).

– Data flow: Exchange of data between the submodels and also between a submodel and the base model is standardized.

– Soft-coding: The model code does not contain "hard-coded" specifications which require a change of the code and recompilation after the model domain or the temporal or spatial resolution is changed. A prominent example is to use height or pressure for parameterizations of vertical profiles, instead of level indices, as the latter have to be changed if the vertical resolution of the base model is adjusted.

– Portability: All submodels are coded according to the language standard of Fortran95 (ISO/IEC-1539-1). The submodel code is free of hardware vendor specific language extensions. In the rare cases where hardware specific code is unavoidable (e.g. to circumvent compiler

deficiencies), it is encapsulated in preprocessor directives.

3 Implementation

To meet the objectives described above, we have developed the Modular Earth Submodel System (MESSy), which comprises.

1. a generalized interface structure for coupling processes coded as so-called submodels to a so-called base model,
2. an extendable set of processes coded as submodels,
3. a coding standard.

The MESSy interface connects the submodels to the base model via a standard interface. As a result, the complete model setup is organized in four layers, as shown in Fig. 4:

1. The Base Model Layer (BML): At the final development state, this layer comprises only a central time integration management and a run control facility for the processes involved. In the transition state (at present) the BML is the domain specific model with all modularized parts removed. For instance, in case of an atmospheric model it can be a GCM.
2. The Base Model Interface Layer (BMIL), which comprises basically three functionalities:
   - The central submodel management interface allows the base model to control (i.e. to switch and call) the submodels.
   - The data transfer/export interface organizes the data transfer between the submodels and the base model and between different submodels. It is furthermore responsible for the output of results (export). Based on the requirements of the model setup, the data can be classified according to their use, e.g. as physical constants, as time varying physical fields, and as tracers (i.e. chemical compounds).
   - The data import interface is used for flexible (i.e. grid independent) import of gridded initial and time dependent boundary conditions.

   The BMIL therefore comprises the whole MESSy infrastructure which is organized in so called generic submodels (see Appendix A).
3. The Submodel Interface Layer (SMIL): This layer is a submodel-specific interface, which collects all relevant information/data from the BMIL, transfers them via parameter lists to the Submodel Core Layer (SMCL, see below), calls the SMCL routines, and distributes the calculated results from the parameter lists back to the BMIL. Since this layer performs the data exchange for the submodel, also the coupling/feedback between different submodels is managed within this layer.
4. The Submodel Core Layer (SMCL): This layer comprises the self-consistent core routines of a submodel (e.g. chemical integrations, physics, parameterizations, diagnostic calculations, etc.), which are independent of the implementation of the base model. Information exchange is solely performed via parameter lists of the subroutines. The output is uniquely determined by the input.

The user interface is implemented by using the Fortran95 namelist constructs, and is connected to the three layers BMIL, SMIL, and SMCL (see Appendix B).

The global switch to turn the submodel on/off is used in the BMIL. These switches for all submodels are set by the run script (see Appendix B).

Submodel-specific data initialization (e.g. initialization of chemical species (=tracers)), and import of data within the time integration (e.g. temporally changing boundary conditions) using the data import interface are handled by the SMIL. Within the SMIL, also the coupling options from the user interface are evaluated and applied, which control the coupling of the submodel to the base model and to other submodels. For instance, the user has the choice to select the submodel input from alternative sources, e.g. from results calculated online by other submodels, or from data provided offline. Therefore, this interface allows a straightforward implementation and management of feedback mechanisms between various processes.

The control interface is located within the SMCL and manages the internal complexity (and with this also the output) of the submodel. It comprises, for instance, changeable parameter settings for the calculations, switches for the choice of different parameterizations, etc.

A directory structure for managing MESSy in a comprehensive model setup is suggested in Appendix C. The basic rules for coding a MESSy-conform submodel are listed in Appendix D.

A model simulation of a typical base model/MESSy setup can be subdivided into three phases (initialization phase, time integration phase, finalizing phase), as shown by a simplified flow chart in Fig. 5.

The main control (time integration and run control) is hosted by the base model, and therefore the base model is also responsible for the flow of the three phases. After the initialization of the base model, the MESSy infrastructure (i.e. the generic submodels) is initialized. At this stage the decision is made which submodels are switched on/off. Next, the active MESSy submodels are initialized sequentially. This initialization is divided into two parts (not explicitely shown in Fig. 5). First, the internal setup of all active submodels is initialized, and second the potential coupling between all active submodels is performed (see Appendix B). After the initialization phase the time integration

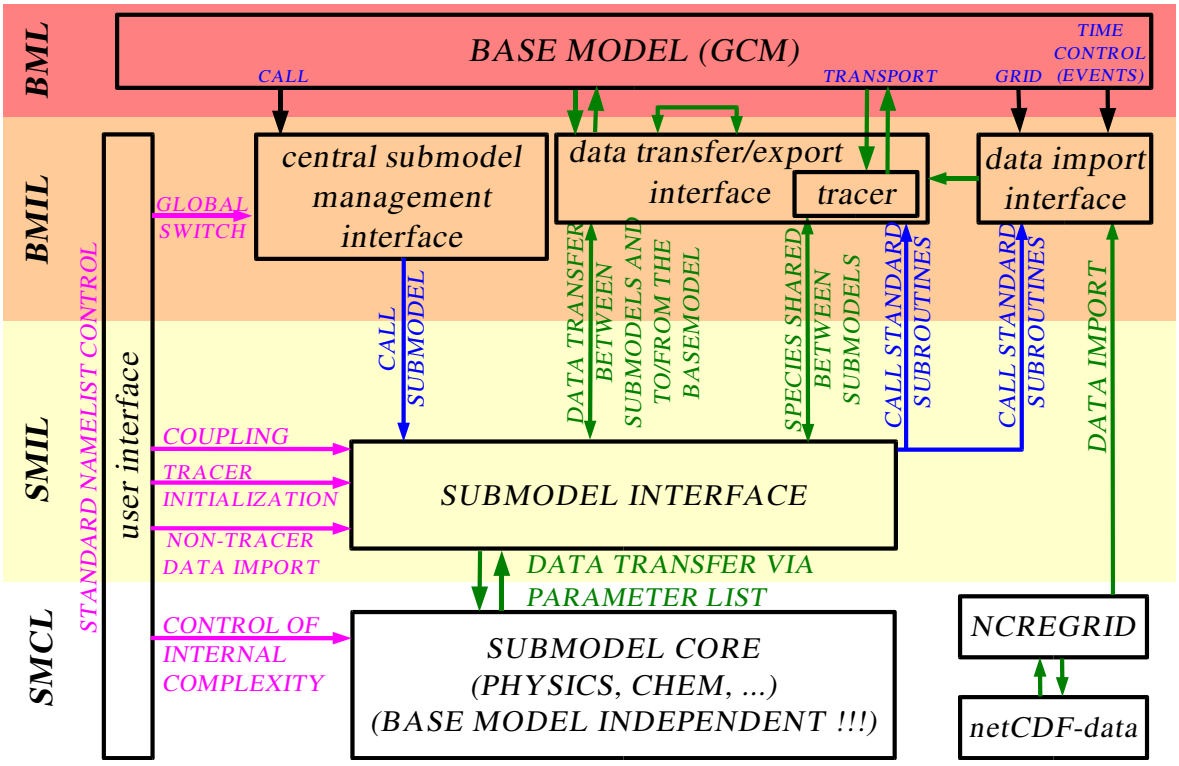

Fig. 4. The four layers of the MESSy interface structure (see text for a detailed description).

(time loop) starts, which is controlled by the base model. All MESSy submodels are integrated sequentially according to the operator splitting concept (see Fig. 3). At the end of the time integration, the MESSy submodels and the MESSy infrastructure are finalized before the base model terminates.

The four layer MESSy-interface structure as presented here can be applied to a variety of different model types with respect to the dimension (e.g. 0-D=box, 1-D=column, 2-D, 3-D), the domain (e.g. global, regional, ocean, atmosphere, land), and then allows the straightforward exchange of processes (i.e. of the submodels in the SMCL). This is shown next.

4 Application

4.1 Box models as the base model

Although most submodels are mainly written for the inclusion into larger regional and global models (e.g. GCMs), the MESSy structure supports and also encourages their connection to a box model, whereby "box model" is defined here as the smallest meaningful entity for a certain process which is running independently of the comprehensive ESM. We have applied the MESSy structure to miscellaneous box models of which two examples are presented.

As a first example, Fig. 6 shows how an atmospheric chemistry submodel can be connected to either a simple box model (here a 0-D model in the classical sense), or to a complex GCM.

Note that exactly the same files of the chemistry submodel are used in both cases. Therefore, box models are an ideal environment for debugging and validating a submodel. While developing the submodel, it can be tested in fast box model runs without the need for expensive global model simulations. Once the submodel performs well, it can directly be included into the GCM without any further changes. A detailed description of the MESSy submodel MECCA (Module for Efficiently Calculating the Chemistry of the Atmosphere) can be found in the companion article of Sander et al. (2005).

A second useful example for a box-model is the re-discretization tool NCREGRID. NCREGRID allows the transformation of 2-D and 3-D gridded geo-data between arbitrary resolutions. A more detailed description of NCREGRID will be published elsewhere (Jöckel, 2005, manuscript in preparation). The same code which is used for the box-model (i.e. the "offline" re-discretization tool NCREGRID) is used in the GCM/ESM as data import interface (which is coded as a generic submodel, see Appendix A) for the import of initial and boundary conditions in arbitrary resolutions.

4.2 GCM as the base model

The MESSy interface has also been successfully connected to the general circulation model ECHAM5 (Roeckner et al., The atmospheric general circulation model

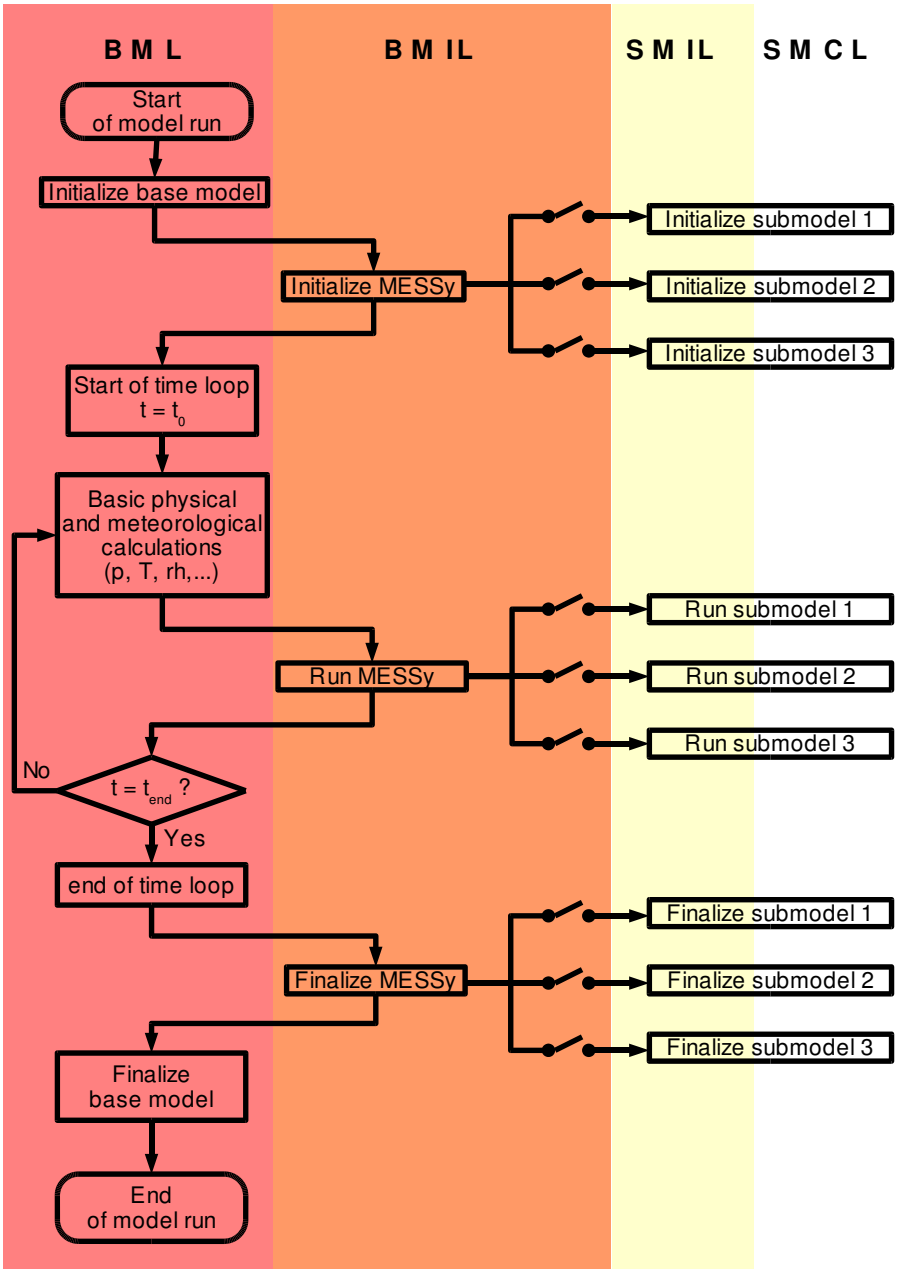

Fig. 5. Idealized flow chart of a typical MESSy setup (see text for details) consisting of three submodels connected to the base model via the MESSy interface. The model simulation can be subdivided into three phases: initialization phase, time integration phase, finalizing phase.

ECHAM 5. PART I: Model description, MPI-Report, 349, 2003 (http://www.mpimet.mpg.de/en/extra/models/echam/mpi_report_349.pdf)), thus extending it into a fully coupled chemistry-climate model. In the electronic supplement (http://www.copernicus.org/EGU/acp/acp/5/433/acp-5-433-sp.zip) details about the specific implementation can be found. This provides exciting new possibilities to study feedback mechanisms. Examples include stratosphere-troposphere coupling, atmosphere-biosphere interactions, multi-component aerosol processes, and chemistry-climate interactions.

MESSy also provides an important tool for model intercomparisons and process studies. As an example, aerosol-climate interactions could be investigated in two model runs in which two different aerosol submodels are switched on. For processes without feedbacks to the base model, it is even possible to run several submodels for the same process simultaneously. For example, the results of two photolysis schemes could be compared while ensuring that they both receive exactly the same meteorological and physical input from the GCM.

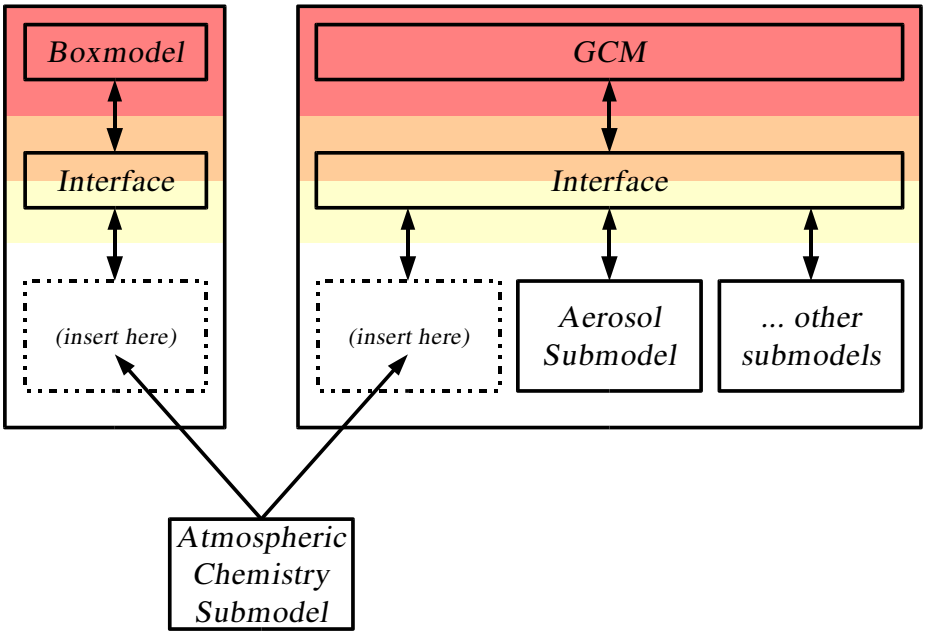

Fig. 6. A MESSy submodel (here an example for atmospheric chemistry integrations) can be coupled to several base models without modifications. The submodel core layer, as indicated by the separated box, is always independent of the base model. Note that at the final development state of MESSy, also the second layer from below (submodel interface layer (SMIL), pale yellow area) is likewise independent of the base model, since it solely makes use of the MESSy infrastructure, for instance the data transfer and export interface. Last but not least, the base model dependence of the second layer from above (base model interface layer, BMIL) is minimized by extensive usage of the MESSy infrastructure. Only well defined connections (as indicated by arrows between BML and BMIL in Fig. 4) are required.

4.3 MESSy submodels

A complete and up to date list of submodels can be found at the MESSy web-site at http://www.messy-interface.org. Detailed description, validation, and application of each submodel will be published elsewhere.

4.4 Future developments

MESSy is an activity that is open to the scientific community following the "open source" philosophy. We encourage collaborations with colleague modelers, and aim to efficiently achieve improvements. The code is available at no charge. For details, we refer to the web-site http://www.messy-interface.org.

Additional submodels and other contributions from the modeling community will be highly appreciated. We encourage modelers to adapt their code according to the MESSy-standard.

5 Discussion

1. Performance: Systematic performance tests are difficult, since fair tests would require a likewise comprehensive and flexible system coded in an "old-fashioned" way, which would be in contradiction. Since data exchange in the MESSy structure is performed within one executable, we expect the performance to be better than that of systems of several specific executables communicating via an additional external coupler.

2. Memory usage: A similar argumentation holds for the memory management: Applying the "classical" coupler-approach, exchanged data needs to be stored in both specific executables communicating to each other, and in addition in the coupler (at least temporarily). In contrast, data in the MESSy approach need to be stored only in one location. All processes (submodels) have access to the same consistent datasets. This is achieved by a pointer-arithmetic based memory-management with minimum overhead.

3. Quick and dirty testing: Of course "quick and dirty" testing is still possible within MESSy, since these tests are mostly related to a specific submodel and therefore independent of the interface. However, we emphasize that these kind of tests are only reasonable during the development and debugging-phase and must not remain in the code for production runs.

4. Scalar and vector architectures: Due to the strict modularization, MESSy allows for a flexible handling on both, vector- and scalar- architectures. Compiler capabilities like automatic vectorization (vector blocking) and inlining can be applied straightforwardly. Last but

not least, for a super-optimization vector- and scalar-code of the same process can coexist and switched on/off depending on the actually used architecture.

6 Conclusions

The transition from General Circulation Models to Earth System Models requires the development of new software technologies and a new software management, since the rapidly increasing model complexity needs a transparent control. The Modular Earth Submodel System (MESSy) provides a generalized interface structure, which allows unique new possibilities to study feedback mechanisms between various bio-geo-chemical processes. Strict compliance with the ISO-Fortran95 standard makes it highly portable to different hardware platforms. The modularization allows for uncomplicated connection to various base models, as well as the co-existence of different algorithms and parameterizations for the same process, e.g. for testing purposes, sensitivity studies, or process studies. The coding standard provides a multi-developer environment, and the flexibility enables a large number of applications with different foci in many research topics referring to a wide range of temporal and spatial scales.

We look forward to receiving interesting contributions from the geosciences modeling community.

Appendix A: The MESSy infrastructure (generic submodels)

The BMIL (Fig. 4) comprises the MESSy infrastructure as described in Sect. 3. This infrastructure is itself coded in the form of generic submodels, i.e. separated into a base model independent part (generic SMCL) and a base model dependent part (generic SMIL). Currently, the MESSy implementation provides the following generic submodels:

- central submodel control interface
  - main_switch: standard user interface for switching the submodels on/off (one global switch per submodel)
  - main_control: provides generalized main entry points for triggering the submodels from the base model
- data transfer/export interface
  - main_tracer: handling of chemical species and their individual properties
  - main_data: data exchange between base model and submodels in both directions
  - main_constants: physical constants and machine precision parameters
- data import interface
  - ncregrid: grid independent input from netCDF files (see http://my.unidata.ucar.edu/content/software/netcdf/index.html for the netCDF format). More information on ncregrid will be published elsewhere (Jöckel, manuscript in preparation, 2005).
- main_tools: common tools shared by several submodels

Appendix B: The MESSy user interface

The MESSy user interface is implemented using Fortran95 namelist constructs. User interaction is required at three stages (in the following `<submodel>` denotes the unique name of a submodel):

1. The run script `xmessy` is controlling the comprehensive model setup. Here, the user selects the submodels to be used for the model run. There is one global switch per submodel for activating it (`USE_<submodel>=T`), or deactivating it (`USE_<submodel>=F`), with the default being 'deactivated'. The run script writes the Fortran95 namelist file `MESSy.nml`, which contains all submodel switches. This namelist file is read by the executable during the MESSy-initialization (central submodel management interface, generic submodel main_switch, see Figs. 4 and 5). Furthermore, the user specifies in the run script which namelist files (e.g., containing predefined setups) should be used to control a specific submodel:

   ```
   NML_<submodel>=...
   NML_<submodel>_T=...
   ```

   The chosen namelist files are copied by the run script into the files `<submodel>.nml` and `<submodel>_t.nml`, respectively, which are the default user interface for a specific submodel.

2. The namelist-file `<submodel>_t.nml` contains the namelists for the data import interface for initialization of chemical compounds (= tracers) during the initialization phase of the model run.

3. The namelist-file `<submodel>.nml` contains the namelists for the submodel operation:
   - The `CTRL`-namelist contains all parameters/switches affecting the internal complexity and flow control of a specific submodel (control interface).

- The CPL-namelist contains all parameters/switches affecting the coupling of a specific submodel to the base model and to other submodels (coupling interface).
- Additional namelists control the data import interface, e.g. for importing time varying boundary conditions at dedicated steps during the time integration phase.
- If the submodel hosts one or more sub-submodels, the namelists for control and coupling of a specific sub-submodel are called CTRL_<sub-submodel> and CPL_<sub-submodel>, respectively.

Appendix C: The MESSy directory structure

All MESSy related files are located in the ./messy subdirectory tree of the base model distribution:

- messy/src: contains the core modules (SMCL files) of the submodels
- messy/<smil>: contains the base model dependent submodel interface modules (SMIL files), whereby <smil> is an appropriate name identifying the used base model
- messy/lib: contains the MESSy-library after successful compilation
- messy/box: contains the source code of the box models (see Sect. 4.1)
- messy/bin: contains the executables of the MESSy-box models after successful compilation
- messy/nml: contains the namelist-files (user interface)
- messy/util: contains utility scripts

The MESSy core modules (SMCL, in messy/src) will be compiled and archived as the library libmessy.a. The MESSy interface modules (SMIL, in messy/<smil> will be compiled and linked with the base model code.

Appendix D: The MESSy coding standard

For the implementation of the MESSy interface, all changes to the base model are coded with "keyhole surgery". This means that changes to the base model are only allowed if they are really needed, and if they are as small as possible. Changes to the base model code are encapsulated in preprocessor directives:

```
#ifndef MESSY
  <original code>
# else
  <changed code for MESSy>
#endif
```

Likewise, additional code is encapsulated as

```
#ifdef MESSY
  <new MESSy code>
#endif
```

Overall, code development follows the following rules:

- Each process is represented by a separate submodel.
- Every submodel has a unique name.
- A submodel is per default switched OFF and does nothing unless it has been switched on (USE_<submodel>=T) by the user via a unique namelist switch in the run script xmessy (see Appendix B).
- Several submodels for the same process (e.g. different parameterizations) can coexist.
- MESSy modules are Fortran95-standard conform (ISO/IEC-1539-1). This can, for example, be checked using the Fortran analyzer "forcheck" (see http://www.forcheck.nl).
- Each submodel consists of two modules (two layers):
  1. submodel core layer (SMCL): A completely self-consistent, base model independent Fortran95 core-module to/from which all required quantities are passed via parameters of its subroutines. Self-consistent means that there are neither direct connections to the base model, nor to other submodels. The core-module provides PUBLIC subroutines which are called by the interface-module, and PRIVATE subroutines which are called internally.
  2. submodel interface layer (SMIL): An interface-module which organizes the calls and the data exchange between submodel and base model. Data from the base model is preferably accessed via the generic submodel SMIL "main_data". The interface module provides a set of PUBLIC subroutines which constitute the main entry-points called from the MESSy central submodel control interface.
- The core module must be written to run as the smallest possible entity (e.g. box, column, column-vector (2-D), global) on one CPU in a parallel environment (e.g. MPI). Therefore, STOP-statements must be omitted and replaced by a status flag (INTENT(OUT)) which is 0, if no error occurs. In the same way, WRITE- and PRINT-statements must only occur in the part of the code which

is exclusively executed by a dedicated I/O processor. This is controlled in the SMIL.

- Data transfer between submodels is performed exclusively via the generic submodel "main_data" within the interface layer. Direct USE-statements to other submodels are not allowed.

- The internal application flow of a submodel is controlled by switches and parameters in the CTRL-namelist; coupling to the base model and/or other submodels is defined via switches and parameters in the CPL-namelist (see Appendix B).

- If the complexity of a submodel requires separation into two or more files per layer (core or interface), shared type-, variable- and parameter-declarations can be located in `messy_<submodel>_mem.f90` files (or `messy_<submodel>_mem_<smil>.f90` files, respectively) which can be USEd by the submodel files within the respective layer. These memory-modules must be used by more than 1 file within the relevant layer, and must not contain any subroutine and/or function.

- The filename of each MESSy file identifies submodel, layer, and type: `messy_<submodel>[_<subsubmodel>][_mem][_<smil>].f90` where `[...]` means "optional", `<...>` means a specific name, `mem` indicates memory-files, and `<smil>` the interface layer modules. Each Fortran module must have the same name as the file it resides in, however with the suffix `.f90` removed.

- MESSy-submodels are independent of the specific base model resolution in space and time. If this is not possible (e.g. for specific parameterizations) or not yet implemented, the submodel needs to terminate the model in a controlled way (via the status flag), if the required resolution has not been chosen by the user.

- A submodel can host sub-submodels, e.g. for different various parameterizations, sub-processes, etc. The namelists of the respective sub-submodel are named according to Appendix B.

- The smallest entities of a submodel, i.e. the subroutines and functions, must be as self-consistent as possible according to:

  - USE-statements specific for a certain subroutine or function must be placed where the USEd objects are needed, not into the declaration section of the module.

  - `IMPLICIT NONE` is used for all modules, subroutines and functions.

  - If a function or subroutine provides an internal consistency check, the result must be returned via an INTEGER parameter (status flag), which is 0, if no error occurs, and identifies the problem otherwise.

- PRIVATE must be the default for every module, with the exception of memory-files. The PUBLIC attribute must explicitely used only for those subroutines, functions, and variables that must be available outside of the module.

- Variables must be defined (not only declared!). The best way to define a variable is within its declaration line, e.g.:

```
INTEGER :: n=0
```

- Pointers need to be nullified, otherwise, the pointer's association status will be initially undefined. Pointers with undefined association status tend to cause trouble. According to the Fortran95 standard even the test of the association status with the intrinsic function ASSOCIATED is not allowed. Nullification of a pointer is preferably performed at the declaration

```
REAL, DIMENSION(:,:), POINTER :: &
  ptr => NULL()
```

  or at the beginning of the instruction block with

```
NULLIFY(ptr)
```

- Wherever possible, ELEMENTAL and/or PURE functions and subroutines must be used.

- Numeric precision is controlled within the code by specifying the KIND parameters. Compiler switches to select the numeric precision (e.g. "-r8") must be avoided.

- Since the dependencies for the build-process are generated automatically, obsolete, backup- or test-files must not have the suffix `.f90`. Instead, they must be renamed to `*.f90-bak` or something similar.

- Any USE command must be combined with an ONLY statement. This makes it easier to locate the origin of non-local variables. (Exception: A MESSy-core module may be used without ONLY by the respective MESSy-interface module.)

Acknowledgements. The authors thank the following people for supporting the MESSy standard, for adapting various submodels to it, and for contributing to specific submodels: S. Brinkop (DLR), C. Brühl (MPI-C), J. Buchholz (MPI-C), M. de Reus (MPI-C), G. Erhardt (DLR), L. Ganzeveld (MPI-C), L. Grenfell (FUB), V. Grewe (DLR), I. Kirchner (FUB), C. Kurz (DLR), S. Metzger (MPI-C), M. Ponater (DLR), A. Pozzer (MPI-C), A. Rhodin (DWD), R. Sausen (DLR), B. Steil (MPI-C), M. Tanarhte (MPI-C), M. Traub (MPI-C), M. van Aalst (MPI-C), J. van Aardenne (MPI-C), R. von Glasow (UHD), R. von Kuhlmann (MPI-C). (DLR=Deutsches Zentrum für Luft- und Raumfahrt=German Aerospace Center, DWD=Deutscher Wetterdienst=German Weather Service,

MPI-C=Max Planck Institute for Chemistry, UHD=University of Heidelberg, FUB=University of Berlin). Support from the Max Planck Computer Center in Garching (RZG) is gratefully acknowledged.

Edited by: M. Dameris

References

Sander, R., Kerkweg, A., Jöckel, P., and Lelieveld, J.: Technical Note: The new comprehensive atmospheric chemistry module MECCA, Atmos. Chem. Phys., this issue, 2005.

Post, D. E. and Votta, L. G.: Computational Science Demands a New Paradigm, Physics Today, 58, 1, 35–41, 2005.